LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

CONF-880455--1

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE: Discrete Event Simulation in the Artificial Intelligence Environment

LA-UR--87-4256

DE88 004296

AUTHOR(S): Douglas J. Roberts, H. W. Egdorf

SUBMITTED TO: 1988 Eastern Simulation Conference

MASTER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO. 836 R4
ST. NO. 2629 5/81

Discrete Event Simulation in the Artificial Intelligence Environment

by

H. W. Egdorf
Douglas J. Roberts

Simulation and Software Development Group, A-5
MS F-602
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Abstract

Discrete Event Simulations performed in an Artificial Intelligence (AI) environment provide benefits in two major areas. The productivity provided by Object Oriented Programming, Rule Based Programming, and AI development environments allows simulations to be developed and maintained more efficiently than conventional environments allow. Secondly, the use of AI techniques allows direct simulation of human decision making processes and Command and Control aspects of a system under study.

An introduction to AI techniques is presented. Two discrete event simulations produced in these environments are described. Finally, a software engineering methodology is discussed that allows simulations to be designed for use in these environments.

1. Artificial Intelligence Environments

Discrete event simulation performed in an Artificial Intelligence environment derives much benefit from special features of those environments. In order to appreciate how these special features assist in simulation modeling, they will be described in enough detail to serve as an introduction for those used to more conventional simulation environments.

1.1. Object Oriented Knowledge Representation

An object oriented programming system is a programming paradigm that provides distinct advantages over conventional procedure oriented programming systems and their associated programming and design paradigms.

An object in an Object Oriented system encapsulates both data and the operations performed on those data. In this respect the object is similar to an abstract data type. However, the Object Oriented programming paradigm goes beyond abstract data types. An Object Oriented system is characterized by the following features.

Object Structure and Message Passing

All objects may contain internal state information known as instance variables. The operations that the object can perform are defined by the set of messages to which the object responds. The procedure used by an object to respond to a message is called a method. An object's methods may reference only its own instance variables. The only interactions between objects take place by way of sending a message and receiving a response to the message.

Inheritance

Objects that share similar instance variables and methods can have the common characteristics extracted into a class. The class is an object that describes the common characteristics shared by either subclasses or instances of the class. The classes form a hierarchy of class-subclass relationships with instances as the leaves of the hierarchy. If the system allows the class-subclass-instance hierarchy to form a general digraph rather than just a tree, the system is said to support multiple inheritance.

The design process of building a class hierarchy from a flat set of objects is similar to the process used to normalize relational database systems.

Late Binding

The Object Oriented system allows response to a message based on the characteristics of the object at the time that the message is passed to the object. This is generally the most difficult of an Object Oriented system to implement upon a procedure oriented language such as C or ADA[1] where the characteristics of the objects are determined at compile-time rather than at run-time.

1.2. Rule Based Knowledge Representation

Much knowledge can be represented by way of if-then rules that examine and modify the state of the knowledge represented in the system. Two main techniques exist to allow a collection of rules to derive new knowledge from existing knowledge.

Forward Chaining

In forward chaining, the antecedents of each rule are examined, and wherever all the antecedents are satisfied the consequents are asserted. For example, in a rule such as

if	(the cargo of MY-TRUCK is CARGO)
	(the state of CARGO is dangerous)
then	(current-activity is tell the fcc about MY-TRUCK)

The forward chaining system would attempt to find all objects CURRENT-TRUCK with a cargo instance

[1] Ada is a registered trademark of the U. S. Government, Ada Joint Program Office.

variable CARGO such that that truck is carrying a dangerous cargo. Wherever such a set of values can be found, it will be asserted that the piece of knowledge in the system known as current-activity will be changed. (Such a change may well cause other rules to become satisfied causing other pieces of knowledge to change.)

Backward Chaining

A backward chaining system works in the other direction. The consequents of each rule are examined, and wherever the conclusion is satisfied with the current knowledge, the antecedents are asserted. For example, in a rule such as

if	(ANIMAL is in class mammal)
then	(the body-cover of ANIMAL is hair)

The backward chaining system will attempt to find values for ANIMAL such that the body-cover of the animal is hair. Wherever such a value is found, the piece of knowledge that the animal is in class mammal will be added to the set of knowledge in the system.

Generally, backward chaining systems are used in deduction and diagnoses systems. We have found the forward chaining system to be most useful in simulation modeling of human decision making.

The ability to capture and model human decision making directly within the rule system is of fundamental importance in constructing maintainable models of systems that depend upon such decision making. The rule system should allow the developer and the maintainer to model knowledge in an English-like way. This is very important to the long-term use of such models.

2. An Example Conventional Simulation Implemented in an AI Environment

A relatively large-scale simulation has recently been completed at Los Alamos in an AI environment. The application, funded by the Albuquerque Area Office of the Department of Energy, was designed to allow DOE to perform long-range planning studies of the Special Nuclear Materials Production Complex in the United States (Figure 1). The following facilities, and their interactive material flows are represented in the model:

- The Rocky Flats Operation
- The Los Alamos TA-55 Operation
- The Oak Ridge Y-12 Operation
- The Fernald FMPC Operation
- The Hanford Operation
- The Savannah River Operation
- The Gaseous Diffusion Plant

The model was developed in KEE[2] and LISP on a Symbolics[3] 3600 computer, and contains approximately 1,100 objects (KEE units). This combination of hardware / software represents today's state-of-the art in AI computing technology. Figure 2 is a graphic of the KEE knowledge base describing the Rocky Flats Plant near Denver, Colorado. Figure 3 shows detail about the object called Building 771[4] at Rocky Flats. Building 771 processes two plutonium-bearing aqueous scrap streams, fast and slow, to recover plutonium metal.

Seven man-years of development were invested in the effort during the one calendar year it took to complete the model. While this particular simulation *could* have been developed in one of the more conventional simulation languages (SLAM, SIMAN, GPSS, SIMSCRIPT), the authors believe that the development time would have been much longer using one of these more conventional languages. In addition, the modularity of the model (because of the object-oriented approach) makes it easier to maintain than had it been built using a procedural approach.

It takes 25 - 30 minutes to complete a ten-year run with all facilities active, or three to four minutes to run just the Rocky Flats facility stand-alone.

3. An Example of a Command, Control, and Communications Simulation

Two years ago, a medium-to-large scale simulation was developed at Los Alamos using KEE on Symbolics hardware to represent the behavior of 1 to N battalion-sized military organizations involved in a combat situation. Each battalion is comprised of tracked vehicles, fuel trucks, reconnaissance vehicles, tow trucks, food, fuel, supplies, and personnel. The personnel include a commander, staff personnel, and support personnel (mechanics, radio operators, gunners, etc.) This class of simulation problem is called Command, Control, and Communication (C^3), and requires a rigorous representation of human decision-making processes to realistically represent the military operation. A few examples of the decision processes modeled in this simulation include:

- Has the threat of detection from enemy radio direction finding systems, human intelligence, overhead surveillance, etc. increased to the point where a move to another covert location is necessary?
- Is it necessary to dispatch a reconnaissance team to gather information about the forward terrain?
- Since it has been determined that a refueling operation is required, to which fuel depot should a re-team be dispatched?
- Given the extant environmental / combat conditions, should the tracked vehicle that was just damaged by a conventional attack be repaired while the battalion waits, should a mobile repair team be dispatched while the rest of the battalion moves on, or should the vehicle be abandoned in place?

In contrast to the example given above of a conventional process simulation, the authors do *not* feel that this C^3 model could have been built and maintained using one of the older, conventional simulation systems. One of the objectives of C^3 modeling is to analyze the effects of different decision-making schemes on the outcome of the simulation. The use of a rule system, where decision processes are defined as data (as compared to compiled If - Then - Else constructs within a

---

[2] KEE (Knowledge Engineering Environment) is a trademark of IntelliCorp of Mountain View, California.

[3] Symbolics is a trademark of Symbolics, Inc.

[4] Note: The processing rates and efficiencies presented in Figure 3 are not the actual data for Building 771 at Rocky Flats.

procedural language) allows this, whereas the older simulation systems do not.

It is important to note that this class of simulations largely performs analysis upon the *doctrine* or *policy* of a system rather than the physical capabilities of the system (although, naturally, the physical processes remain an important part of the simulation). This new ability is the most important characteristic of this new simulation environment when compared to more conventional simulation environments.

4. A Software Engineering Methodology

Given an understanding of the Artificial Intelligence environment characterized by object oriented programming and rule based programming, and given an understanding of a sample discrete event simulation produced in this environment, the software engineering methodology used to develop these simulations is now presented.

4.1. Actors

A set of requirements for a simulation will always include information regarding the situation to be simulated and the analysis desired about this situation. Given these requirements, the first step in designing a simulation is to characterize all active entities in the situation as *Actors*. The definition of an *Actor* here is quite broad, encompassing any and all active entities in the situation to be simulated. The concept of an *Actor* contains a fair bit of anthropomorphism.

Any actor will have three sets of characteristics.

Assets and Attributes
: Each *Actor* may have certain items in its possession. For example, a fuel truck will have a certain amount of fuel on hand as an asset.

Physical Capabilities
: Each *Actor* may be able to exercise an ability to react in some way. For example, a fuel truck will be able to change location, provide fuel to some other *Actor*, and replenish its fuel assets from a central fuel depot.

Cognitive Capabilities
: Each *Actor* may be able to exercise command and control, or decision making abilities. For example, a fuel truck (or more specifically its driver) can examine the current situation and follow some doctrine or decision making process to determine how to proceed. Perhaps the fuel assets of the fuel truck have fallen below a level that would prevent future, scheduled refuelings to occur. In this case, the driver may elect to divert to a central facility for re-fueling.

  It is the ability to directly model these cognitive capabilities that differentiates between simulations in Artificial Intelligence environments and those in more conventional environments.

An **activity** in the simulation will be a representation of an *Actor* exercising one of its capabilities. An exercise of an *Actor's* cognitive capabilities is no different than an exercise of an *Actor's* physical capabilities although the representation of the exercise of a physical capability will differ from the representation of the exercise of a cognative capability within the model's implementation.

Finally, each activity is represented by an event that represents the instant in time when an activity either begins or completes. Some activities happen instantaneously and are modeled as a single event.

4.2. Actors as Objects

Once the system to be simulated is described in terms of *Actors* and their associated assets, physical capabilities, and cognitive capabilities, the simulation can be constructed by modeling each *Actor* as an object in the object oriented programming system. The object's instance variables are used to represent assets of the *Actor*. The events that represent the start and end of the activities representing the *Actor's* physical and cognitive capabilities become methods of the object. All events in the simulation are implemented as messages passed to the objects representing the *Actors* in the drama.

A major advantage of the Artificial Intelligence environment comes into play at this point. The methods representing the exercise of a physical capability will be encoded in a procedure oriented language (e.g. Lisp, C, FORTRAN) in the same way as in conventional simulation environments. However, rather than encoding the exercise of a cognitive capability in the procedure oriented language (the only option in the conventional simulation environment), the Artificial Intelligence environment provides a forward-chaining or a backward-chaining (or both!) rule system that is ideal for capturing and modeling the decision making processes of the exercise of an *Actor's* cognitive capabilities. Additionally, the rules defining these cognitive capabilities are data, *not* compiled code.

5. Comparison of Environments

The primary difference between discrete event simulation models developed in the Artificial Intelligence environment and such models developed in conventional programming environments is the ability to use a rule system to model decision making. This leads to a more comprehensive view of an activity in the simulation produced in the Artificial Intelligence environment.

In the conventional environment, emphasis is placed on modeling the exercise of physical capabilities. This leads to a model of sequences of activities represented as a sequence of events

- Begin activity
- End activity.

With the ability to model the exercise of cognitive capability, each activity is modeled as the event sequence

- Begin Cognitive activity to determine next action.
- End Cognitive activity
- Begin Physical activity.
- End Physical activity.
- Begin assessment activity to determine result of physical activity.
- End assessment activity.

The new sub-activities that represent the cognitive and assessment portions allow modeling of command and control systems, and allow a model to be built that performs analysis of the decision making process itself, rather than just performance analysis of a physical system.

These new sub activities allowing modeling of decision making are only practical given the knowledge modeling tools provided by the rule systems within the Artificial Intelligence environments. The argument can be made that the

Artificial Intelligence environments provide no fundamental capability over the more conventional environments for simulation. This argument is usually based on examples of a rule-based system implemented as branches within a procedural programming framework. This argument is valid as far as it goes, however, it does not go far enough. Either environment is Turing-equivalent in the computational sense. Any function computable in a procedure-oriented system can also be computed in an object-oriented system and vice-versa. The difference is one of ease of development and maintenance over the life-cycle of the simulation.

The conventional simulation environment forces an over-emphsis on the representation of the physical processes of a system to be modeled. With no tools available to directly model cognative processes, much of current simulation design is spent determining how to model decision making processes stochasticly and removing any decision making process that cannot be modeled in this way. While it is a tribute to Statisticians that so much human decision making can be modeled stochasticly, conventional simulation environments do not provide tools to allow any other sort of modeling of the decision making process. Often, systems analysis requires the study of non-stochastic decision-making processes. The advantage of the Artificial Intelligence environment is that this sort of knowledge can now be modeled directly within the rule system. Many simulations can now be implemented that were not possible before. In particular, the decision making doctrine of a system can now be modeled directly in the same way that physical processes were modeled before the advent of the new simulation environments.

It is our belief that only trivial decision modeling can be performed by embedding the doctrinal knowledge within branches of a procedural oriented language where the model must be maintained over the analysis lifetime of the model. The Artificial Intelligence environments provide both increased development productivity over conventional simulation environments and the new ability to model human decision making processes.

Figure 1.

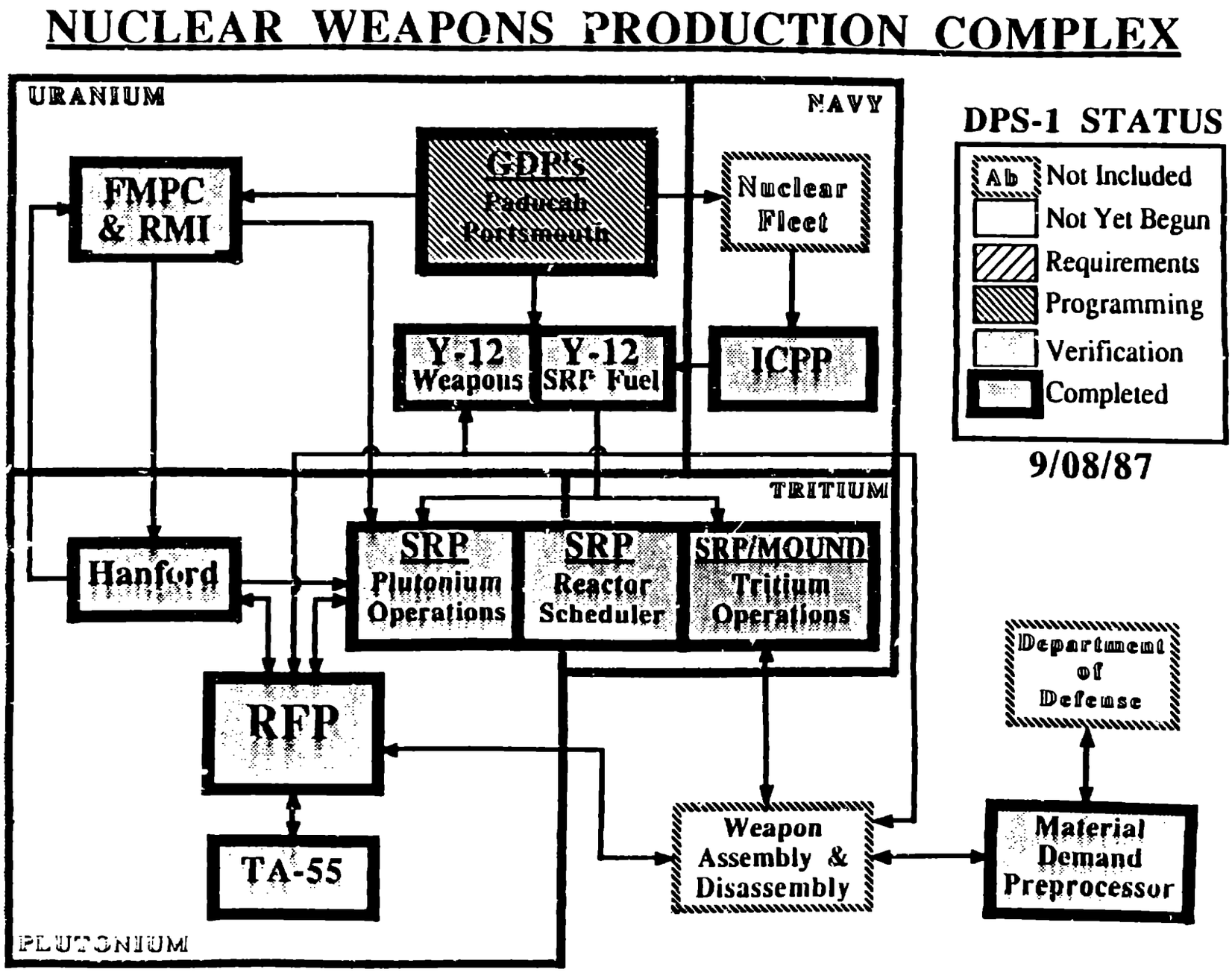

NUCLEAR WEAPONS PRODUCTION COMPLEX
URANIUM
NAVY
FMPC & RMI
GDPs
Paducah
Portsmouth
Nuclear Fleet
Y-12 Weapons
Y-12 SRP Fuel
ICPP
TRITIUM
Hanford
SRP Plutonium Operations
SRP Reactor Scheduler
SRP/MOUND Tritium Operations
RFP
TA-55
PLUTONIUM
Department of Defense
Weapon Assembly & Disassembly
Material Demand Preprocessor
DPS-1 STATUS
Ab Not Included
Not Yet Begun
Requirements
Programming
Verification
Completed
9/08/87

Figure 2.

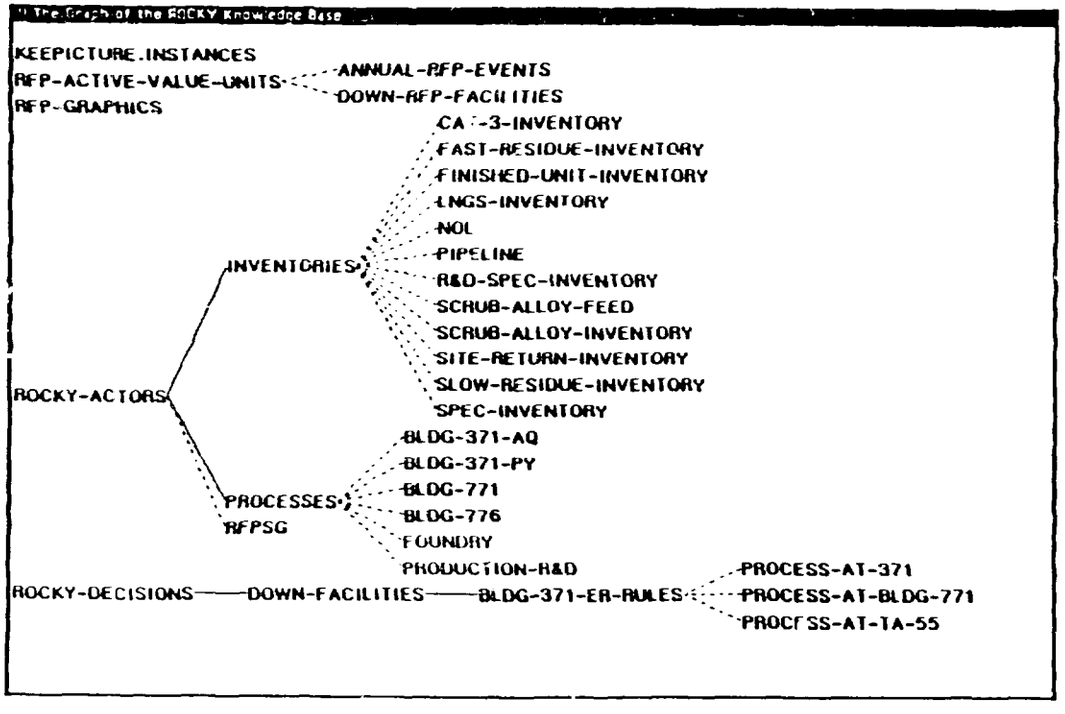
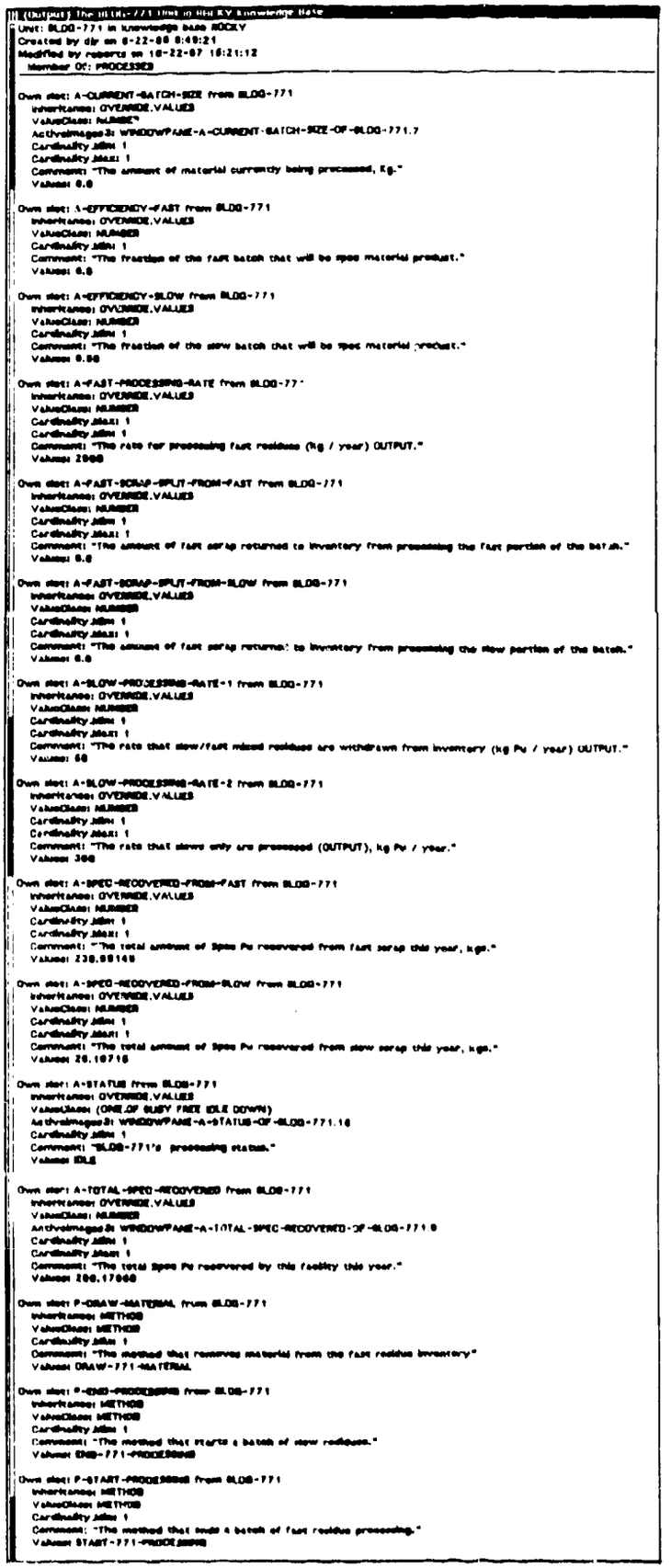

The Graph of the ROCKY Knowledge Base
KEEPICTURE.INSTANCES
RFP-ACTIVE-VALUE-UNITS
RFP-GRAPHICS
ANNUAL-RFP-EVENTS
DOWN-RFP-FACILITIES
CAT-3-INVENTORY
FAST-RESIDUE-INVENTORY
FINISHED-UNIT-INVENTORY
LNGS-INVENTORY
NOL
PIPELINE
R&D-SPEC-INVENTORY
SCRUB-ALLOY-FEED
SCRUB-ALLOY-INVENTORY
SITE-RETURN-INVENTORY
SLOW-RESIDUE-INVENTORY
SPEC-INVENTORY
INVENTORIES
ROCKY-ACTORS
PROCESSES
RFPSG
BLDG-371-AQ
BLDG-371-PY
BLDG-771
BLDG-776
FOUNDRY
PRODUCTION-R&D
ROCKY-DECISIONS
DOWN-FACILITIES
BLDG-371-ER-RULES
PROCESS-AT-371
PROCESS-AT-BLDG-771
PROCESS-AT-TA-55

Figure 3.


(Output) The BLDG-771 Unit in HEALY Knowledge Base
Unit: BLDG-771 in knowledge base HEALY
Created by dlr on 8-22-88 8:49:21
Modified by reborn on 10-22-87 16:21:12
Member Of: PROCESSES
Own slot: A-CURRENT-BATCH-SIZE from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
ActivelmageOf: WINDOWPANE-A-CURRENT-BATCH-SIZE-OF-BLDG-771.7
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The amount of material currently being processed, Kg."
Values: 0.0
Own slot: A-EFFICIENCY-FAST from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Comment: "The fraction of the fast batch that will be spec material product."
Values: 0.0
Own slot: A-EFFICIENCY-SLOW from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Comment: "The fraction of the slow batch that will be spec material product."
Values: 0.50
Own slot: A-FAST-PROCESSING-RATE from BLDG-77
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Max: 1
Cardinality.Min: 1
Comment: "The rate for processing fast residues (Kg / year) OUTPUT."
Values: 2000
Own slot: A-FAST-SCRAP-SPLIT-FROM-FAST from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The amount of fast scrap returned to inventory from processing the fast portion of the batch."
Values: 0.0
Own slot: A-FAST-SCRAP-SPLIT-FROM-SLOW from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The amount of fast scrap returned to inventory from processing the slow portion of the batch."
Values: 0.0
Own slot: A-SLOW-PROCESSING-RATE-1 from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The rate that slow/fast mixed residues are withdrawn from inventory (kg Pu / year) OUTPUT."
Values: 60
Own slot: A-SLOW-PROCESSING-RATE-2 from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The rate that slows only are processed (OUTPUT), kg Pu / year."
Values: 300
Own slot: A-SPEC-RECOVERED-FROM-FAST from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The total amount of Spec Pu recovered from fast scrap this year, kgs."
Values: 238.99148
Own slot: A-SPEC-RECOVERED-FROM-SLOW from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The total amount of Spec Pu recovered from slow scrap this year, kgs."
Values: 20.10715
Own slot: A-STATUS from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: (ONE.OF BUSY FREE IDLE DOWN)
ActiveImageOf: WINDOWPANE-A-STATUS-OF-BLDG-771.10
Cardinality.Min: 1
Comment: "BLDG-771's processing status."
Values: IDLE
Own slot: A-TOTAL-SPEC-RECOVERED from BLDG-771
Inheritance: OVERRIDE.VALUES
ValueClass: NUMBER
ActiveImageOf: WINDOWPANE-A-TOTAL-SPEC-RECOVERED-OF-BLDG-771.9
Cardinality.Min: 1
Cardinality.Max: 1
Comment: "The total Spec Pu recovered by this facility this year."
Values: 298.17868
Own slot: P-DRAW-MATERIAL from BLDG-771
Inheritance: METHOD
ValueClass: METHOD
Cardinality.Min: 1
Comment: "The method that removes material from the fast residue inventory"
Values: DRAW-771-MATERIAL
Own slot: P-END-PROCESSING from BLDG-771
Inheritance: METHOD
ValueClass: METHOD
Cardinality.Min: 1
Comment: "The method that starts a batch of slow residues."
Values: END-771-PROCESSING
Own slot: P-START-PROCESSING from BLDG-771
Inheritance: METHOD
ValueClass: METHOD
Cardinality.Min: 1
Comment: "The method that ends a batch of fast residue processing."
Values: START-771-PROCESSING